**Bezier Curves Documentation**

**How to run?**

* This program is very simple webGL program. Please go to “./Source” and open index.html. You should able to run it without installing any external libraries.

**User Interface: 2 Buttons**

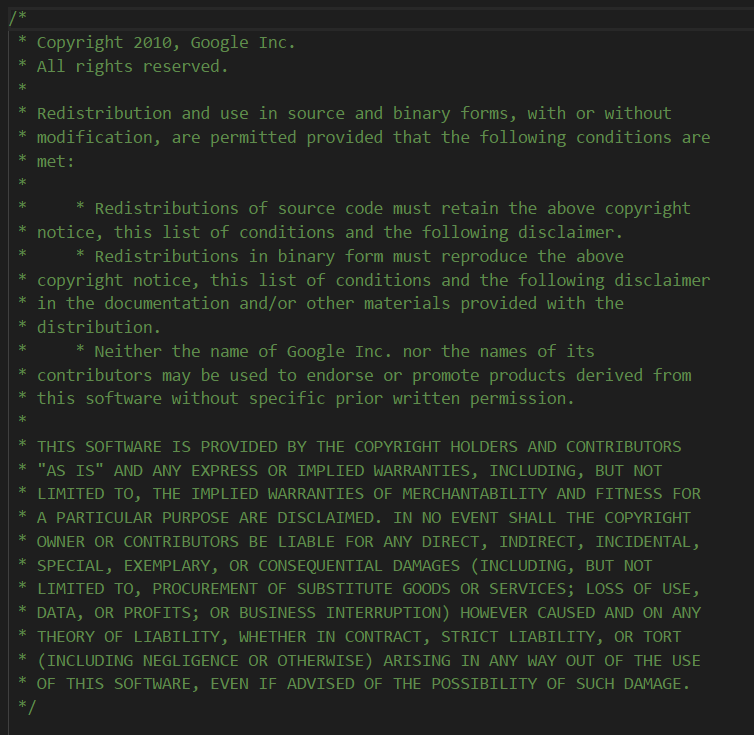
1. Bezier Curvature: this button will render a near-perfect curvature.
2. Wireframe: this button will render a wireframe which is drawn by connecting each control point.

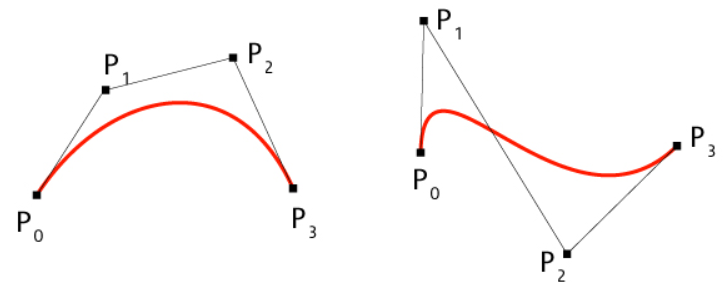
**Hardcoded Data**

* The data are in “data.js” file inside “./Source” directory.
* Currently, I have hardcoded “teapot” and “teaspoon”. You may uncomment either one to render whichever object you want.

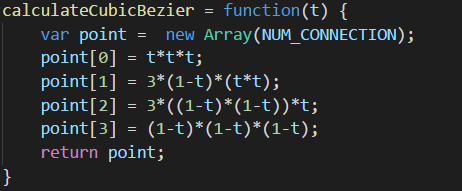
**What I did**

* As you observe, this program is very simple WebGL program. “./Helper” directory is an open source code that I have got from Google.



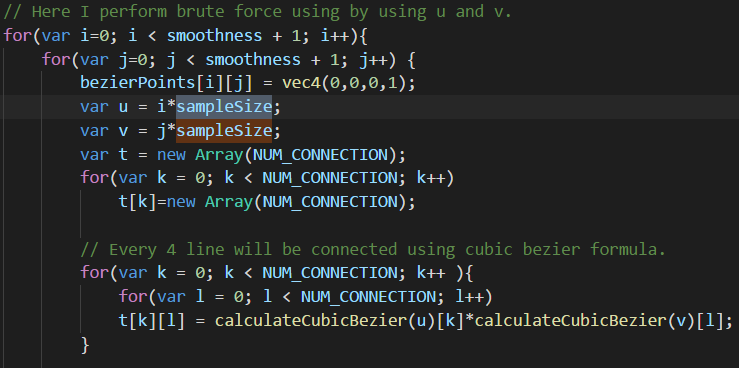
The main functionalities are in “bezier.js” file. What I have done is to use cubic Bezier formula to draw the curve. If you look at the global variables, you may see “NUM\_CONNECTION = 4”. This means that I am using every 4 point to connect like the figure below.

If you look at “calculateCubicBezier” function inside “bezier.js” file, it’s pretty obvious what I am calculating.

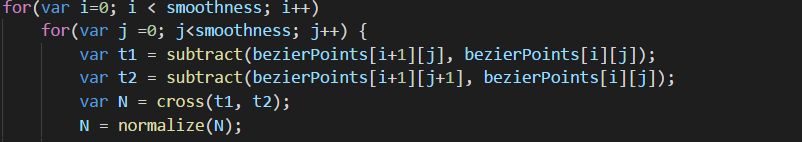


Now, using this function, I can find the point, but how did I actually draw the smooth curve using these points? If you look at the global variables section again, there exists “smoothness = 100”. This represents how smooth I want the curves to be. Using “smoothness = 100”, I calculate the sample size by “1.0 / smoothness”. So this means that I am calculating up to 0.001 floating point, which should be pretty accurate curve.

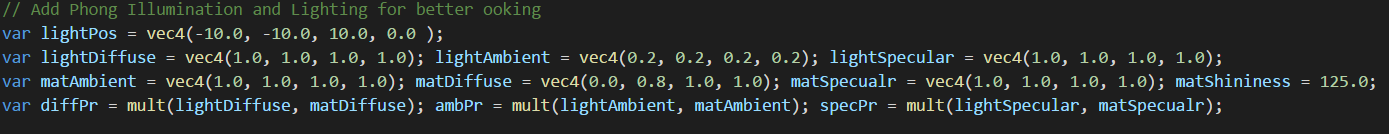
Using this smoothness and sample size, I have used a brute force method to draw the curvatures. I have decided to use the brute force method because it made more sense to me. I increment u and v gradually by sample size and find each point using Bezier formula, and connect each line to draw a curvature.



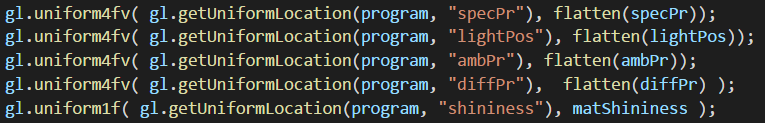
Then, I have filled the patches using shaders. In order to fill the patches with colours, I had to calculate the normals. I calculate the normal for every point ( the points that have been calculated by “calculateCubicBezier” function ) in order to render more realistic object.



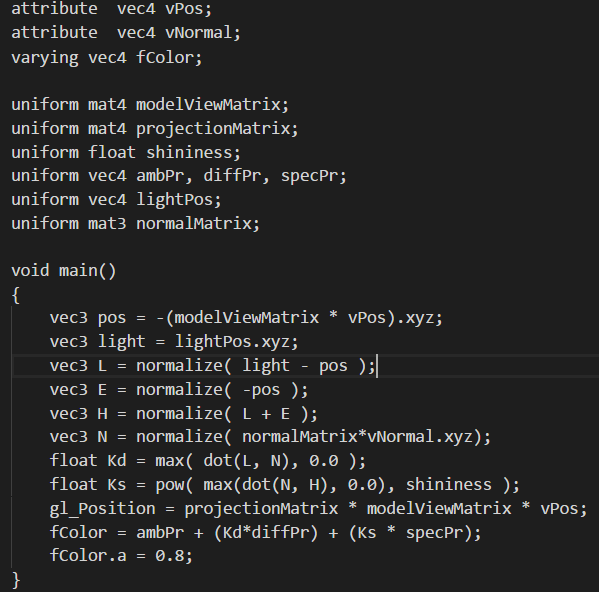
Then, I have used Phong Illumination using these declarations.



These diffuseProduct, ambientProduct, specularProduct, shininess and etc. are then passed onto the vertex shader.



This vertex shader code is in “index.html” file. As you see, I am basically calculating initiating the scene with lights, 3d space and phong illumination.



Then, of course there is fragment shader as well.

